Vol.2016-SE-194 No.4
2016/11/17

OSS 開発におけるパッチの特徴量を用いた再投稿要求の予測

安藤聡志[†1] 伊原彰紀[†2] 関浩之[†1]
平尾俊貴[†2] 則兼卓人[†2] 松本健一[†2]

OSS 開発では，変更されたソースコード（パッチ）の振る舞いの確認，欠陥混入の調査などを目的に，開発者（レビューア）が検証（コードレビュー）を行う．パッチに欠陥が混入していた場合は，レビューアがパッチ開発者に再修正を要求する．本論文は，パッチの再投稿を未然に防ぐために，変更されたソースコードの特徴量に基づき，パッチの再修正をレビューアから要求されるか否かを予測する手法を提案する．本論文では，ケーススタディとして Qt プロジェクトに投稿されたパッチ 21,668 件を対象として提案手法の評価実験を行った．2 種類の予測モデルを用いたケーススタディの結果，パッチの特徴量に基づく予測モデルの適合率は 0.345，再現率は 0.578，F 値は 0.431 となり，ファイルの特徴量に基づく予測モデルの適合率は 0.413，再現率は 0.249，F 値は 0.310 となった．

1. はじめに

ソフトウェア開発プロセスでは，ソースコードの新規開発，および，既存のソースコードを改変したファイル（パッチ）を，製品に統合する前に，ソースコードの振る舞い，欠陥の有無，可読性などを開発者（レビューア）が検証する．商用ソフトウェア開発では，検証作業としてテストケース，テストコードが用意されていることが多いが，オープンソースソフトウェア (OSS) 開発では，テストコードが十分に準備されていないことが多く，レビューアによる検証がソフトウェアの品質を確保する上で重要な作業である[3]．

昨今，大規模ソフトウェア開発におけるパッチ検証では，Gerrit [⋆1] や Review Board [⋆2] をはじめとするレビュー管理システムが使用されている．レビュー管理システムは，投稿されたパッチに対して，複数のレビューアが各々検証した評価結果について合意形成を行い，パッチの採択，再投稿，不採択が決定する．しかし，OSS プロジェクトにパッチ投稿する開発者は必ずしも高い開発技術を持っているとは限らず，すべてが高品質であるとは限らない．本論文が分析対象とする Qt プロジェクトでは，プロジェクトに投稿されたパッチ 70,705 件のうち 28,276 件（約 39.9%）は再修正が必要となり，開発者がパッチを再投稿している．再投稿が必要となるパッチは，1 度のレビューで採択されるパッチと比べて約 2 日長い．従って，パッチの再投稿は，パッチの品質向上が期待される一方で，開発効率の低下に影響する．

Weißgerber ら[1] は，大規模 OSS プロジェクト FLAC と OpenAFS を対象に，採択されるパッチの特徴を調査し，パッチの行数（変更行数）が少ないほど採択される可能性が高いことを示している．これまでに我々は，Qt プロジェクトを対象に同様の調査を行い，Weißgerber らと同じ結果を確認している．本論文では，ソースコード行数以外の代表的な特徴量（サイクロマチック数，コメント行数など）によるパッチ再投稿率の調査も行った．そして，パッチの再投稿を未然に防ぐことを目的に，パッチの特徴量に基づき，パッチが再投稿されるか否かを予測する手法を提案する．本論文では，ケーススタディとして Qt プロジェクトに投稿されたパッチ 21,668 件を対象として提案手法の予測性能を実験的に評価した結果を述べる．

続く 2 節では，一般的なコードレビュープロセスと実施した予備調査について述べる．3 節では，提案手法について説明し，4 節では，ケーススタディを実施した結果について述べる，最後に 5 節で本論文のまと

†1 名古屋大学 大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†2 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

⋆1 https://www.gerritcodereview.com/
⋆2 https://www.reviewboard.org/

ⓒ 2016 Information Processing Society of Japan

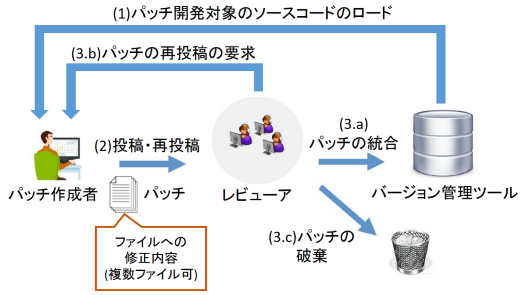

図 1　コードレビュー手順の図

めと今後の課題について述べる．

2. コードレビュープロセス

Gerrit や ReviewBoard などのコードレビュー管理システムは，ソースコード中にレビューアの指摘をタグ付けするなどの機能を備えており，コードレビューを容易にしている．また，ソースコードのバージョン管理ツールとの連携することができ，コードレビュー管理システムから直接，バージョン管理ツールに最新バージョンを投稿することも可能である．図 1 は，パッチ作成からパッチの検証終了までの一連の手順である．

(1) 開発者はパッチ開発対象のソースコードを自身のローカル環境にコピーする．
(2) 開発者はソースコードを変更し，パッチをコードレビュー管理システムに投稿する．
(3) レビューアが投稿されたパッチをレビューする．レビューアがパッチを採択と判断した場合，パッチはバージョン管理システムに統合される．パッチの修正が必要である場合は，開発者に再修正を要求し，不採択であれば破棄される．

手順 (2) で作成したパッチに不備があり手順 (3) で再修正が必要となった場合，手順 (2) に戻ってパッチの修正を行う．そのため，作成したパッチに不備が出続ける場合，コードレビューに費やす時間は増大する．本論文はこのような状況を回避するため，手順 (2) の段階で，手順 (3) でパッチが再投稿となるか否かを予測し，品質の高いパッチのみを投稿できるようにすることを目的とする．

コードレビュー管理システムには，パッチの修正内容を反映する前のファイル（パッチ適用前のファイル），パッチの修正内容を反映した後のファイル（パッチ適用後のファイル）が保存されている．

2.1 予 備 調 査

本論文では，一度のレビューで採択されるパッチの特徴を理解するために，ソースコードの代表的な特徴量（サイクロマチック数，コメント行数など）によるパッチ再投稿率の予備調査を Qt プロジェクトを対象に行う．Qt は UI フレームワークであり，ソフトウェア開発企業 Digia の一部門によって開発されているが，Qt プロジェクトは OSS 開発のように不特定多数の開発者が貢献している．Qt プロジェクトではレビュー管理システム Gerrit を用いてレビューデータを管理している．本論文では，著者らが Qt プロジェクトのリポジトリから収集したデータセットを使用した．データセットにはパッチ 21,668 件の修正対象となったファイル，コードレビュー結果，パッチ投稿者のコメント，各パッチごとの修正対象となったファイルの修正内容が変更されたか（ファイルが再投稿の原因となったか）が登録されている．また，本予備調査では，ソースコードの特徴量の計測に用いるテクマトリックス社のソースコード解析ツール「Understand Ver. 4.0」 [*1] において計測可能な 34 種類の特徴量 [*2] を用いる．ただし，ソースコード行数，コメント行数，際クロマチック数，宣言された関数の数を除く特徴量の値は，ファイルの変更前後で変化がわずかであったため，本論文では対象外とする．

予備調査では，ソースコード変更前後のファイルから 4 つの特徴量を計測し，その変化量（絶対値）を分析する．

(1) パッチで修正対象となっているソースコードの特徴量の変化量でパッチをソートする．
(2) パッチ 1000 件ごとに特徴量の変化量の平均，および，再投稿となったパッチの割合を計算する．
(3) 横軸に平均の特徴量の変化量，縦軸に再投稿となったパッチの割合をとってグラフを描画する．

図 2，図 3，図 4，図 5 は，それぞれソースコード行数，コメント行数，サイクロマチック数，宣言された関数の数に対する再投稿されたパッチの割合を示す．ソースコード行数，コメント行数，サイクロマチック数，宣言された関数の数の変化量は，変化量の絶対値が大きいほど再投稿される割合が高い．ただし，コメント行数，サイクロマチック数，宣言された関数の数の変化量については，特徴量が変化していないパッチが多い．

予備調査では，ソースコード行数，コメント行数，サイクロマチック数，宣言された関数の数の特徴量（絶対値）が大きくなるにつれ，再投稿される確率が上昇することがわかった．本論文では，これらの特徴量を用いて予測モデルを提案する．

*1 http://understand.techmatrix.jp/
*2 http://understand.techmatrix.jp/features/metrics/metrics_list/

ⓒ 2016 Information Processing Society of Japan

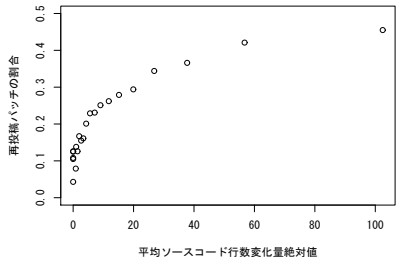

図 2 ソースコード行数変化ごとの再投稿割合 (絶対値)

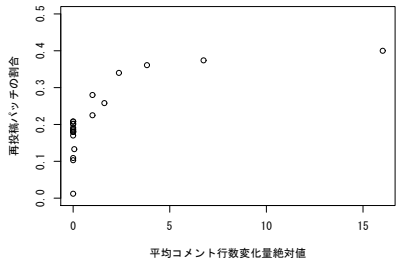

図 3 コメント行数変化ごとの再投稿割合 (絶対値)

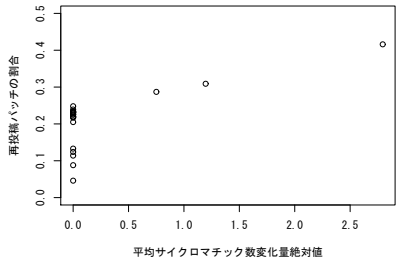

図 4 サイクロマチック数変化ごとの再投稿割合 (絶対値)

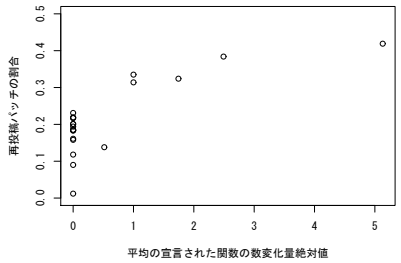

図 5 宣言された関数の数変化ごとの再投稿割合 (絶対値)

3. 提 案 手 法

パッチの再投稿が必要となるか否かを予測するモデルを提案する．モデルは，パッチ適用前のソースコードの特徴量とパッチ適用後のソースコードの特徴量と

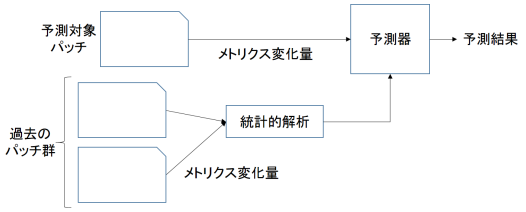

図 6 パッチの特徴量に基づく予測モデル生成の手順

の差を入力値とし，パッチが再投稿となるか否かを出力結果とする．ただし，パッチは一度に複数のファイルを変更されていることがある．本論文では，同時に変更されたファイル群の特徴量の変更量をまとめて入力値とし，パッチが採択されるか否かを予測するモデル（パッチの特徴量に基づく予測モデル）と，ファイルごとの変更量を入力値とし，パッチが採択されるか否かを予測するモデル（ファイルの特徴量に基づく予測モデル）を提案する．

3.1 パッチの特徴量に基づく予測モデル

パッチは，同一目的のために複数のファイルを変更することがある．本モデルは，パッチ適用前のファイル群の特徴量の合計値と，パッチ適用後のファイル群の特徴量の合計値との差を入力値とし，そのパッチが再投稿となるか否かを予測する．本モデルの構築には，過去に投稿されたパッチの適用前後のファイル群の特徴量，および，再投稿の有無を用いる．図 6 は，予測モデル構築方法の概略図を示す．

図 6 では予測器を 1 つのみ生成しているが，本モデルで良い予測精度が期待できるのは，特徴量と再投稿率の関連において全てのパッチが同じ傾向を持つという場合である．例えば，1,000 行のソースコードを 10 行変更する場合と，20 行のソースコードを 10 行変更する場合とで再投稿となる確率がほぼ同じである場合のみ妥当である．本論文では，ソースコード規模に対する変更量を考慮するために，コード行数の規模に応じて異なる予測モデルを構築（規模分類）する．

さらに，パッチが再投稿が必要となる理由は，ソースコードの内容に限らない[4]．変更目的に応じて再投稿されることもある．本論文では，パッチの変更目的の違いによる予測モデルの構築（意味分類）にも取り組む．

従って，本論文では 3 種類のパッチの分類方法による予測モデルを構築する．

- 分類なし
- ソースコードの規模，具体的にはコード行数で分類（規模分類）
- パッチ投稿者のコメントに特定の単語が含まれて

ⓒ 2016 Information Processing Society of Japan

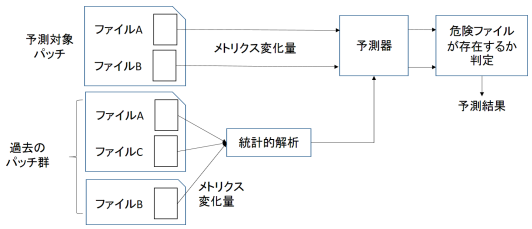

図 7 ファイルの特徴量に基づく予測モデル生成の手順

いるかで分類（意味分類）

規模分類では修正前の修正対象となるファイルの規模による分類を行う．具体的には，コード行数で k-means 法 ($k = 8$) を用いてパッチをクラスタリングする．意味分類ではパッチがバグ修正を目的としているかで分類を行う．具体的には，パッチ投稿者のコメントが “**error**”，“**fix**”，“**run**”，“**stop**” のいずれかを含む場合はパッチがバグ修正を目的としていると判断し，そうでなければパッチがバグ修正を目的としていないと判断する．予測の際は予測対象を分類し，予測対象に合った予測器を使用して予測を行う．

3.2 ファイルの特徴量に基づく予測モデル

理想的なオブジェクト指向ソフトウェア開発ではモジュール間の結合度が低くなるため，あるファイルに対する修正は他のファイルに影響を与えることが少ない．従って，修正対象のファイルそれぞれについて特徴量の計測とそのファイルが再投稿の原因となる（以降，そのようなファイルを危険ファイルと呼ぶ）か否かを予測するモデルを構築する．修正対象のファイルに危険ファイルが 1 つでも存在したらそのパッチが再投稿すると予測し，危険ファイルが存在しなければ再投稿する必要がないと予測する．

本モデルにおいて，ファイルが危険ファイルであるかの予測にはファイルを訓練データとして生成された予測器を使用する．本モデルでは図 7 のような手順で予測器の生成と再投稿となるか否かを予測する．

ファイルの特徴量に基づく予測モデルにおいても同様に 3 種類のモデル構築（分類なし，規模分類，意味分類）を行う．なお，意味分類は，同じパッチで修正対象となっているファイルは同じパッチ投稿者のコメントを用いて分類を行う．

4. 評 価 実 験

4.1 実験データ

本論文では予備調査と同じ Qt プロジェクトのデータセットを用いて実験を行う．データセットに登録されているパッチのうちの約 80%（17,369 件）を訓練データとし，残り 20%（4,299 件）を予測データとして実験を行う．

データセット中のソースコードの特徴量の計測にはソースコード解析ツール「Understand Ver.4.0」を使用する．計測および予測では予備調査と同じ特徴量の絶対値を使用する．予測器は，多重線形回帰分析を用いて構築する．予測結果の評価には，適合率，再現率，F1 値を用いる．

表 1 パッチの特徴量に基づく予測モデル結果

	分類なし	規模分類	意味分類
適合率	0.345	0.302	0.342
再現率	0.578	0.621	0.569
F 値	0.431	0.406	0.427

表 2 ファイルの特徴量に基づく予測モデルの結果

	分類なし	規模分類	意味分類
適合率	0.413	0.406	0.415
再現率	0.249	0.218	0.253
F 値	0.310	0.284	0.314

4.2 実 験 結 果

パッチの特徴量に基づく予測モデルの評価結果を表 1 に示す．ファイルの特徴量に基づく予測モデルの評価結果を表 2 に示す．

ファイルの特徴量に基づく予測モデル（以下，ファイルモデル）は，変更ファイルのうち，1 件以上の危険ファイルが存在した場合に再投稿と予測するため，再投稿と予測される確率がパッチの特徴量に基づく予測モデル（以下，パッチモデル）に比べて高いと考えられる．しかし，パッチモデルは，ファイルモデルに比べて，適合率が低いことがわかった．予測モデル構築に使用した訓練データを調査したところ，ファイルモデルは単一のソースコードに対して危険ファイルか否かを予測しており，パッチモデルに比べて再投稿と予測される確率が低かった（パッチモデルにおける再投稿の割合: 25.3%，ファイルモデルにおける危険ファイルの割合: 18%）ことが原因と考えられる．

規模分類，および，意味分類を実施した場合にかかわらず，予測精度に差はほとんど見られなかった．規模分類については，今後，規模によるパッチ再投稿の割合を調査する．

また，ファイルの特徴量に基づく予測モデルにおいて，より粒度の細かい分類としてファイル名による分類（名前分類）を行った．名前分類は，同じファイルはパッチによる修正を受けても，ファイルの特徴量の変化量と危険ファイルである割合の関連が変化しないという仮定が成り立つ時に予測精度の向上が期待できると考えられる．名前分類も他の分類と同様の手順により予測器の生成と評価実験を行った．ただし，同じファイルが複数のパッチで修正対象となっている場合

ⓒ 2016 Information Processing Society of Japan

表 3 名前分類を実施したファイル特徴量に基づく予測モデルの結果の比較

	分類なし	規模分類	意味分類	名前分類
適合率	0.422	0.430	0.412	0.410
再現率	0.249	0.235	0.245	0.236
F 値	0.310	0.284	0.314	0.299

表 4 予測が成功したパッチと失敗したパッチの平均再投稿回数

	パッチモデル	ファイルモデル
予測成功	2.72	3.05
予測失敗	1.78	2.07

は少なく，ファイルによっては予測器を生成できるだけの十分な訓練データが集まらないため，本論文では訓練データとなるファイルが 5 件以上存在する場合のみ予測器の生成と評価実験を行った．実験結果を表 3 に示す．名前分類も他の分類と同様に予測精度に大きな影響を与えないことがわかった．

コードレビューにおいて，再投稿は必ず一度のみ行われるわけではなく，パッチの品質によっては複数回の再投稿が行われる．ここまでは一度でも再投稿となったかどうかのみを見てきたが，予測が成功したパッチと失敗したパッチの再投稿となった回数に注目する．再投稿予測が成功したパッチと失敗したパッチについて，モデルごとに平均の再投稿回数を計算したものを表 4 に示す． パッチモデル．ファイルモデルの両方で再投稿予測が成功したパッチ群の方が失敗したパッチ群よりも再投稿回数が多い．これは，特徴量の変化量の絶対値が大きいものは一度の修正で不備をなくしきることができない可能性が高いことを表すと考えられる．

5. まとめと今後の課題

本論文では，パッチの再修正を回避することを目的として，変更ファイルの特徴量に基づく再投稿予測モデルを提案した．

予備調査の結果，パッチ適用時のソースコード行数，コメント行数，サイクロマチック数，宣言された関数の数の変化量の絶対値と再投稿率に関連があることが分かった．本論文では，これらの 4 つのパッチ適用前後の特徴量の変化量に基づいてパッチが再投稿となるか否かを予測する手法として，パッチの特徴量を用いる場合とファイルの特徴量を用いる場合の 2 種類の手法を提案した．2 種類の手法の予測精度を実験により評価した結果，ファイルの特徴量を用いる場合の方が適合率が高くなることが分かった．これらの手法の予測精度の改善のために予測の前にパッチおよびファイルの分類を行った結果，どちらの予測手法でも分類による予測精度の向上は見られなかった．

予測精度の向上のための今後の課題としては，特徴量の変化量の絶対値と再投稿率の関連を示すグラフの形状を踏まえ，予測器生成手法を多重線形回帰分析から変更することが考えられる．多重線形回帰分析は比例関係となっている対象の予測に対して有効であるため，準備調査でのグラフから考えると，対数を利用した分析手法の方が適している可能性がある．また，予備調査でのグラフによれば，特徴量の変化量の絶対値が大きくても再投稿率は 50% を超えない．よって，再投稿となるか否かの二値ではなく，どれだけ再投稿になりやすいかの割合をユーザーに提示する方が実用性が高い可能性があるため，それについても検討が必要である．

謝辞 本研究の一部は，頭脳循環を加速する戦略的国際研究ネットワーク推進プログラムによる助成を受けた．

参 考 文 献

1) Peter Weißgerber, Daniel Neu, Stephan Diehl, “Small Patches Get In!,” *Proceedings of the International Working Conference on Mining Software Repositories (MSR’08)*, pp.67-76, 2008.
2) 安藤 聡志, 平尾 俊貴, 伊原 彰紀, 松本 健一, 関 浩之, OSS 開発におけるソースコード静的解析手法を用いたパッチ検証手法の提案, ウィンターワークショップ 2016・イン・逗子 論文集, pp.49-50, 2016.
3) ． Pavneet Singh Kochhar, Tegawende F. Bissyande, David Lo, and Lingxiao Jiang, “An Empirical Study of Adoption of Software Testing in Open Source Projects,” Proceedings of the International Conference on Quality Software (QSIC’13), pp.29-30, 2013.
4) Yida Tao, Donggyun Han, and Sunghun Kim “Writing Acceptable Patches: An Empirical Study of Open Source Project Patches,” Proceedings of the International Conference on Software Maintenance and Evolution (ICSM’14), pp.271-280, 2014.

ⓒ 2016 Information Processing Society of Japan